Localização de Palavras II

Renato Cordeiro ferreira

17 de junho de 2013

1 Introdução

Neste exercício-programa, tínhamos como objetivo criar uma estrutura que manipulasse as saídas geradas pelo programa CoreNLP, disponibilizado pela Universidade de Stanford[1]. O CoreNLP é um software de processamento de linguagem natural que oferece o recurso de "lematização": para cada palavra presente num texto, geramos o correspondente "lema" - a origem da palavra.

Para este programa, foram utilizadas as estruturas de tabelas de hash com solução de conflitos por *probing linear* e por encadeamento, de modo a obter algoritmos de complexidade constante (O(1)) para inserções e remoções na maioria dos casos.

2 Arquivos

O presente EP é composto dos seguintes arquivos:

- Item.h
- list.c
- list.h
- arne.c
- arne.h
- enc.c
- enc.h

[1]Disponível em `http://nlp.stanford.edu/software/corenlp.shtml`

- lp.c
- lp.h
- word.c
- word.h
- lemma.c
- lemma.h
- getline.c
- getline.h
- main.c
- Makefile

Cada um deles apresenta algumas particularidades:

2.1 item

O item generalizado que serve como configuração para as implementações das tabelas de símbolos. Neste EP, fizemos uma modicação em relação ao anterior, transformando as chaves de tipos **void *** para **char ***.

2.2 list

Uma lista generalizada, implementada com o auxílio de ponteiros do tipo **void *** e com tipos de 1ª classe. É possível usá-la como uma lista genérica para itens de qualquer tipo.

2.3 arne

Tabela de símbolos implementada com o uso de **ARNEs** (Árvores Rubro-Negro Esquerdistas). Estas árvores foram generalizadas para a utilização como tipo de 1ª classe e para aceitarem qualquer tipo de estrutura (por meio de ponteiros **void ***).

As árvores rubro-negras foram baseadas no código disponibilizado na página do professor Yoshiharu (llrb.c) e modificados para aceitar diferentes tipos de chaves. A função de definição da chave precisa ser especificada

juntamente com as funções “less”, “eq” e o tipo “NULLitem” na função “STinit()“.

Para este EP, os arquivos das ARNEs foram mantidas. Entretanto, suas implementações não são utilizadas.

2.4 enc

O arquivo correspondente a enc.c implementa um protótipo de tabela de símbolo com várias funções de hash baseadas na implementação do Prof. Yoshiharu e generalizadas como tipos de 1ª classe para poder ser usado entre as tabelas T1 e T2. A solução das colisões se dá por meio de encadeamentos (criação de listas ligadas) que auxiliam a colocar itens de mesma chave. O hash usado foi simples e a tabela se redimensiona sempre que a proporção N/M ¿ 1/2 (N o total de objetos, M o total de espaços na tabela).

2.5 pl

O arquivo correspondente a pl.c implementa um protótipo de tabela de símbolo com várias funções de hash baseadas na implementação do Prof. Yoshiharu e generalizadas como tipos de 1ª classe para poder ser usado entre as tabelas T1 e T2. A solução das colisões se dá por meio de probing linear (saltos dentro do vetor de chaves) que auxilia a colocar itens de mesma chave distribuídos no vetor. O hash usado foi simples e a tabela se redimensiona sempre que a proporção N/M ¿ 1/2 (N o total de objetos, M o total de espaços na tabela).

2.6 word

Módulo relacionado a funções de manipulação da tabela de símbolos que armazena palavras. Contém estruturas que guardam uma lista de sentenças, o lema e a própria palavra (por meio de ponteiros para um buffer) que auxiliam a imprimir, contar e acessar os dados presentes na tabela de símbolos.

2.7 lemma

Um módulo que utiliza as estruturas de lista e tabela de símbolos para poder construir as funções necessárias para o acesso, contagem e impressão de dados relacionados aos lemas.

2.8 getline

O arquivo 'getline', retirado das notas de aula do prof. Yoshiharu, foi utilizado como um módulo para leitura de strings de tamanho desconhecido. Com ele, foi possível produzir um buffer que armazenasse o texto lido.

2.9 main

Arquivo principal do texto que contêm a interface com o usuário (via linha de comando e opções de execução) e a leitura (em fase de pré-processamento) do texto passado como parâmetro ao programa.

2.10 makefile

Arquivo para a compilação dos arquivos. Sua estrutura é de propósito geral e permite compilar projetos diversificados com diretórios.

3 Considerações finais

Dada a generalização das estruturas propostas por esta implementação do exercício-programa, o custo do uso de ponteiros **void *** durante quase todo o processo diminuiu a eficiência do pré-processamento e carregamento das estruturas de dados.

Isso não impediu, porém, que o acesso e impressão dos dados fossem realizados de forma eficiênte, dado que os algoritmos de complexidade logarítmica (O(n)) inerentes às ARNEs mantiveram-se assimptoticamente ótimos para a estrutura.